REMARKS/ARGUMENTS

The specification is amended to reduce the length of the abstract. No new matter has been added by way of the amendments to the specification, and no patentable subject matter was deleted from the specification.

Claims 1-20 are pending in the present application. Claims 1, 11, 19 and 20 are amended. Support for amendments to claims is as follows: with regard to claims 1, 11 and 19, support may be found in the language of the claims. With regard to claim 11, further support for a, "recordable type medium," may be found on page 28, 2nd paragraph. With regard to claim 20, further support may be found in the preamble of claim 19. Reconsideration and allowance of the claims is respectfully requested.

I. Objection to Claims

The Examiner has stated that claim 20 is objected to as being an improper dependent claim. Applicants have amended claim 20 accordingly.

II. 35 U.S.C. § 101

The Examiner rejected claims 11-18 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. Applicants have amended independent claim 11 to positively claim, "a computer usable recordable type medium having computer executable instructions tangibly embodied thereon." This language complies with 35 U.S.C. § 101. Therefore, this rejection is overcome.

III. 35 U.S.C. § 112, Second Paragraph

The Examiner rejected claims 1-20 under 35 U.S.C. § 112, second paragraph, as indefinite. Without admitting that the original language of the claims was indefinite, Applicants have amended claims 1, 11 and 19 to replace the phrase, "may be," with "is capable of being." This language is definite. Therefore this rejection has been overcome.

IV. 35 U.S.C. § 102, Anticipation

The Examiner rejected claims 1, 3, 7, 10, 11, 13, 15, 18 and 19 under 35 U.S.C. § 102 as anticipated by *Ramme*, U.S. Patent Application Publication No. 2003/0093420, Method and System for Retrieving Sharable Information Using a Hierarchically Dependent Directory Structure, May 15, 2003 (hereinafter *Ramme*). This rejection is respectfully traversed. Regarding claim 1, the Examiner states:

> "Regarding claim 1, *Ramme* teaches a method comprising: A) "determining if the web application includes a reference to at least one shared web module that may be incorporated into a plurality of web applications; and B) "identifying a

location of the at least one shared web module; and C) "logically merging the at least one shared web module with web modules of the web application, if any, to generate a logically merged web application"
With, respect to A), the Examiner notes that *Ramme* teaches the claimed features of "determining if the web application includes a reference to at least one shared web module that may be incorporated into a plurality of web applications" as **[For information which is sharable among two or more entities such as applications which have module files stored in module directories, the sharable information is stored in one of the module directories, and a link to that directory is stored in each of the directories of the modules that share the information]** (see Ramme, paragraph [0013] on page 1).
With respect to B), the Examiner notes that *Ramme* teaches the claimed features of "identifying a location of the at least one shared web module" as [**A search may be performed by first searching the directory of the module. If the file is not found, then the link is used to access the directory upon which the module's directory depends, so that it may be searched in a similar manner. The search may continue iteratively from one directory to another, until the file is located]** (see Ramme, paragraph [0031] on page 3).
With respect to C), the Examiner notes that *Ramme* teaches the claimed features of "logically merging the at least one shared web module with web modules of the web application" as **[For example, this particular arrangement of the files and links imposes a hierarchical structure on the directory storage, in the form of a directed acyclic graph]** (see Ramme, paragraph [0035] together with Figure 1 elements (120,122, 124, 126))."

Office Action August 16, 2007 pp 7-8. (emphasis in original)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims. Claim 1 as amended is as follows:

1. A method of generating a logically merged web module for a web application, comprising:

determining if the web application includes a reference to at least one shared web module that is capable of being incorporated into a plurality of web applications;

identifying a location of the at least one shared web module; and

logically merging the at least one shared web module with web modules of the web application, if any, to generate a logically merged web application.

Under the standards of *In re Bond*, *Ramme* does not anticipate claim 1 because *Ramme* does not

teach the claimed features of, "... determining if the web application includes a reference to at least one shared web module that is capable of being incorporated into a plurality of web applications;" "identifying a location of the at least one shared web module;" and "logically merging the at least one shared web module with web modules of web applications." The Examiner believes otherwise citing various portions of *Ramme*.

Regarding the feature of, "...determining if the web application includes a reference to at least one shared web module that is capable of being incorporated into a plurality of web applications..." the Examiner believes the following portion of *Ramme* teaches this claimed feature:

> "For information which is sharable among two or more entities such as applications which have module files stored in module directories, the sharable information is stored in one of the directories, and a link to that directory is stored in each of the directories of the modules that share the information."

Ramme, paragraph [0013], page 1.

The cited portion of *Ramme* teaches linking directory structures to form a chain of directories. However, *Ramme* does not determine if a web application includes a reference; rather, *Ramme* provides a reference to a directory from a directory, as shown in Figure 1 of *Ramme*. Thus, *Ramme* fails to disclose the claimed feature. Accordingly, *Ramme* does not anticipate claim1.

With regard to the claimed feature of, "identifying a location of the at least one shared web module..." the Examiner believes the following portion of *Ramme* teaches this claimed feature:

> "...a search may be performed by first searching the directory of the module. If the file is not found, then the link is used to access the directory upon which the module's directory depends, so that it may be searched in a similar manner. The search may continue iteratively from one directory to another, until the file is located."

Ramme, paragraph [0031], page 3.

The cited portion of *Ramme* teaches providing a link enabling a search for a file to traverse directory structures until the file is found. However, *Ramme* fails to provide identification of a location of at least one shared module as recited in the claim. *Ramme* instead teaches starting a search in the initial directory of the module and, if the file is not found, traversing to the next directory using the link provided. Because *Ramme* does not have a reference to the location of the desired file, *Ramme* must follow the defined directory chain to search for the file. *Ramme* therefore fails to disclose the claimed feature. Accordingly, *Ramme* does not anticipate claim1.

Regarding the claimed feature of, "...logically merging the at least one shared web module with web modules of the web application..." the Examiner believes the following portion of *Ramme* teaches this claimed feature:

> "The link may be a real link, such as a Unix link, or it may be implemented as a file with a directory specification stored in association with the directory storage area.
> The link file may include all necessary information for accessing another software component, i.e., may include information on a dependent directory storage area and optionally, may include information on software components available at the dependent directory storage area."

Ramme, paragraph [0057], [0058] page 5.

The cited portion of *Ramme* teaches a logical tree structure for directories. *Ramme* teaches linking directory structures together, rather than merging modules within a web application as recited in the claim. For example, *Ramme* shows in Figure 1 links between directory structures forming a hierarchical structure, rather than merging modules in an application as recited in the claim. In fact, *Ramme* has nothing to do with merging application modules. *Ramme* therefore fails to disclose the claimed feature. Accordingly, *Ramme* does not anticipate claim1.

The remaining claims all contain features similar to those presented in claim 1. Accordingly, under the standard of *In re Bond*, the rejection of claims 1-20 has been overcome.

V. 35 U.S.C. § 103, Obviousness

A) The Examiner rejected claims 2, 4, 5, 9, 12, 17 and 20 under 35U.S.C. 103(a) as obvious over *Ramme* in view of *Spotswood et al.*, US Patent Application Publication No. 200410255293, System and Method For Using A Classloader Hierarchy to Load Software Applications, December 16, 2004 (hereafter *Spotswood*). This rejection is respectfully traversed. The Examiner states:

> "Regarding claim 2, most of the limitations of this claim have been noted in the rejection of claim 1. However, *Ramme* does not expressly disclose the claimed feature of "loading the logically merged web application into a web container". Spotswood et al, from the same or similar field of endeavors, discloses the application server constructs the application container with the application components in the order in which they were retrieved, resulting in a hierarchical classloader structure in the newly constructed application (see Spotswood, paragraph [0076] on page 5 together with Figure 6). Thus, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references to achieve the claimed feature of "loading the logically merged web application into a web container" (as cited in claim 2). Such combination would have permitted teaching's Spotswood to allow Ramme's to better control over the reloading and namespace separation of individual modules, including EJB's (see Spotswood, paragraph [0019])."

Office Action August 16, 2007 page 11

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The prior art reference (or references when combined) must teach or suggest all the claim

limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In determining obviousness, the scope and content of the prior art are... determined; differences between the prior art and the claims at issue are... ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). "Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue." *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). "*Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.* Id. (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006))."

Claim 2, as originally provided, states:

> The method of claim 1, further comprising:
> loading the logically merged web application into a web container.

The Examiner failed to state a *prima facie* obviousness rejection against claim 2 because the proposed combination does not teach or suggest all the features of claim 2. Specifically, the proposed combination does not teach the features of, "... determining if the web application includes a reference to at least one shared web module that is capable of being incorporated into a plurality of web applications;" "identifying a location of the at least one shared web module;" and "logically merging the at least one shared web module with web modules of web applications."

The Examiner believes otherwise, citing various portions of *Spotswood.* However, *Spotswood* fails to teach or suggest the claimed features missing from the teaching of *Ramme. Spotswood,* as in paragraph [0020], is directed toward, "...allowing individual software modules to be reloaded in memory without forcing other modules to be reloaded at the same time." Thus, the teachings of *Spotswood* are unrelated to the above-identified features of claim 2.

As shown above, *Ramme* does not teach or suggest the claimed features. Therefore combining the module reloading teaching of *Spotswood* cannot produce the results of the claimed invention. Accordingly, the proposed combination of references, considered as a whole, does not teach or suggest the feature of claim 2. Hence, the examiner failed to state a *prima facie* obviousness rejection against the claim 2.

Regarding the feature, "loading the logically merged web application into a web container," the Examiner believes the following portion of *Spotswood* teaches this claimed feature:

> "FIG. 6 illustrates the method used in one embodiment of the invention by the application server to construct the application container. An initial step 200, which can be performed at any time, is to allow the software developer to the edit the application configuration file, which will then determine the hierarchy of modules to be loaded. Then, in step 202, the application server initially receives a request for loading application components, usually from a client machine although applications may be executed on the same machine on which the application server itself resides. In step 204, the application server parses the configuration or control file (in one embodiment the application deployment descriptor file, for example in a WebLogic environment the weblogic-application.xml file) which contains the classloader hierarchy. The classes, modules, and other application components specified within the control file are recognized by the application server. In step 206, the application server proceeds to retrieve the specified application components from a computer readable medium (memory, disk, or other storage) in a manner consistent with the tag layout (i.e. the hierarchy) in the control file. In step 208, the application server then constructs the application container with the application components in the order in which they were retrieved, resulting in a hierarchical classloader structure in the newly constructed application.

Spotswood paragraph [0076] page 5.

The cited portion of *Spotswood* teaches the construction of an application container using a configuration file to load application components in the order they are retrieved. In contrast, rather than components as taught by *Spotswood*, the claimed invention performs, "...loading the logically merged web application into a web container..." in the handling of a logically merged application. This feature is entirely distinct from a "logically merged web application," as claimed. Accordingly, *Spotswood* fails to teach or suggest this claimed feature. Additionally, no other portion of *Spotswood* teaches or suggests this claimed feature.

Further, *Spotswood* also fails to provide the claimed elements missing from *Ramme*. Accordingly, *Ramme* in combination with Spotswood, considered as a whole, does not render obvious claim 2.

With regard to claim 4, claim 4, as originally filed, states:

> "The method of claim 1, wherein the web application is an enterprise archive (EAR) and wherein the logically merged web application is a logically merged EAR."

Regarding the feature, "the web application is an enterprise archive (EAR) and wherein the logically merged web application is a logically merged EAR," the Examiner believes the following portion of *Spotswood* teaches this claimed feature:

> "In a typical application server environment, for example a WebLogic Server implementation, classloading is centered on the concept of an application. An application is normally packaged in an Enterprise Archive (EAR) file containing application classes. Everything within an EAR file is considered part of the same

> application. The following may be part of an EAR or can be loaded as standalone applications:"

Spotswood paragraphs [0049], on page 3.

Spotswood teaches a concept centered on the application and is not related to the claimed feature of, "...logically merged web application is a logically merged EAR." *Spotswood* isolates application as stated in paragraph [0054], and therefore does not share across applications. Thus *Spotswood* is devoid of disclosure regarding the feature of claim 4. Further *Spotswood* also fails to provide the claimed elements missing from *Ramme*. Accordingly, *Ramme* in combination with Spotswood, considered as a whole, does not render obvious claim 4.

With regard to claim 5, claim 5, as originally filed, states:

> "The method of claim 1, wherein the at least one shared web module includes at least one of a web archive (WAR) file, an enterprise java bean (EJB) archive file, and a resource archive (RAR) file."

Regarding the feature, "the at least one shared web module includes at least one of a web archive (WAR) file, an enterprise java bean (EJB) archive file, and a resource archive (RAR) file," the Examiner believes the following portion of *Spotswood* teaches this claimed feature:

> "In a typical application server environment, for example a WebLogic Server implementation, classloading is centered on the concept of an application. An application is normally packaged in an Enterprise Archive (EAR) file containing application classes. Everything within an EAR file is considered part of the same application. The following may be part of an EAR or can be loaded as standalone applications:
> An Enterprise JavaBean (EJB) JAR file;
> A Web Application WAR file; and/or,
> Resource Adapter RAR file."

Spotswood paragraphs [0049], [0050], [0051], [0052], on page 3.

Spotswood teaches a concept centered on the application, and is not related to the claimed feature of, "...at least one shared web module includes at least one of a web archive (WAR) file, an enterprise java bean (EJB) archive file, and a resource archive (RAR) file." *Spotswood* isolates application as stated in paragraph [0054], and therefore does not share across applications. Thus, *Spotswood* is devoid of disclosure regarding the feature of claim 5. Further *Spotswood* also fails to provide the claimed elements missing from *Ramme*. Accordingly, *Ramme* in combination with Spotswood, considered as a whole, does not render obvious claim 5.

With regard to claim 9, claim 9 as originally filed states;

> "The method of claim 2, wherein the container uses one or more application program interfaces (APIs) to identify a path to the at least one shared web module and loads the at least one shared web module when loading the logically merged web application."

The Examiner believes otherwise citing various portions of *Spotswood*. However, *Spotswood* also fails to teach or suggest the claimed features missing from the teaching of *Ramme*. *Spotswood,* as in paragraph [0054], is directed toward, "...every application receives its own classloader hierarchy." The parent of this hierarchy is the system classpath classloader. This "isolates applications so that application A cannot see the classloader or classes of application B."

As shown above, *Ramme* does not teach or suggest the claimed features; therefore combining the EJB interface teaching of *Spotswood* cannot produce the results of the claimed invention. Accordingly, the proposed combination of references, considered as a whole, does not teach or suggest the claimed features.

Regarding the feature, "the container uses one or more application program interfaces (APIs) to identify a path to the at least one shared web module and loads the at least one shared web module when loading the logically merged web application," the Examiner believes the following portion of *Spotswood* teaches this claimed feature:

> "Since EJB classes are invoked through an interface, it is possible to load individual EJB implementation classes in their own classloader. In this manner, these classes can be reloaded individually without having to redeploy the entire EJB module."
>
> "After the redeploy command, the developer can then provide a list of files relative to the root of the exploded application that they want to update. This might be the path to a specific element (as above), or a module, or any set of elements and modules. For example, the developer can use the following command to redeploy another ejb."

Spotswood paragraphs [0078], [0080] page 5.

The teaching of *Spotswood* may provide an interface, but fails to teach the use of a shared web module as recited in claim 9. Further, *Spotswood* isolates applications to prevent sharing as recited in paragraph [0054], and therefore does not provide or suggest the features as claimed.

The Examiner failed to state a *prima facie* obviousness rejection against claim 9 because the proposed combination does not teach or suggest all the features of claim 9. Specifically the proposed combination does not teach the features of, "uses one or more application program interfaces (APIs) to identify a path to the at least one shared web module and loads the at least one shared web module when loading the logically merged web application."

In addition, the Examiner has failed to establish a *prima facie* obviousness rejection against claims 2, 4, 5, 9 12, 17 and 20 because no reason exists under the standards of *KSR Int'l.* to combine the references, considered as a whole, in the manner suggested by the Examiner. No reason exists to combine the references because *Spotswood* teaches away from the claims.

A reference may be said to "teach away" from the claimed invention when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant. *In re Gurley*, 27 F.3d 551, 553, 31 U.S.P.Q.2D 1130, 1131 (Fed. Cir. 1995).

Spotswood teaches away from the claims because the addition of *Spotswood* to the proposed combination would vitiate the entire purpose of *Ramme,* as well as be contrary to the claimed invention.. *Ramme* is directed toward, "...shared use of modules or software components is enabled through directory links..." while *Spotswood* teaches at paragraph [0054] of page 3, "...isolates applications so that application A cannot see the classloaders or classes of application B." On their face, these two purposes are contradictory.

Because *Spotswood* teaches away from claims 2, 4, 5, 9 12, 17 and 20, and *Ramme,* no motivation exists to combine *Spotswood* and *Ramme*, in accordance with *KSR Int'l.*, as proposed by the Examiner. Accordingly, the Examiner has failed to state a *prima facie* obviousness rejection against claims 2, 4, 5, 9 12, 17 and 20.

B) The Examiner rejected claims 6, 8, 14 and 16 under 35 U.S.C. 103(a) as obvious over *Ramme* in view of *Croney et al.*, US Patent Application Publication No. 200410255233 Al, Utilizing Common Layout and Functionality of Multiple Web Pages, December 16, 2004, (hereafter *Croney*). This rejection is respectfully traversed. The Examiner states:

> Regarding claim 6, most of the limitations of this claim have been noted in the rejection of claim 1. However, *Ramme* does not expressly disclose the claimed feature of "determining a priority associated with the at least one shared web module" and "resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any". *Croney* et al, from the same or similar field of endeavors, discloses the server system can be configured to control which of multiple master pages are utilized to form a resulting page based upon predetermined criteria that are satisfied by the occurrence of one or more events or circumstances (see *Croney et al.*, paragraph [0056]). *Croney et al.* further discloses that the modules of the server system can be configured with appropriate computer-executable instructions to recognize predetermined criteria that imposed by a master page, client system, or server system and to recognize when these predetermined criteria have been satisfied (see *Croney et al.*, paragraph [0058]). Thus, it would have been obvious to. one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references to achieve the claimed features of "determining a priority associated with the at least one shared web module" and "resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any" (as cited in claim 6). Such combination would have permitted teaching's *Croney* to allow *Ramme*'s to avoid duplicating of code and content between

multiple web pages and unnecessarily filling up the storage with duplicative content (see *Croney*, paragraph [0007]).

Office Action August 16, 2007 pp 14-15

Claim 6, as originally provided, states:

> "The method of claim 1, wherein logically merging the at least one shared web module with web modules of the web application includes:
> determining a priority associated with the at least one shared web module; and
> resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any."

The Examiner failed to state a *prima facie* obviousness rejection against claim 6 because the proposed combination does not teach or suggest all the features of claim 6. Specifically the proposed combination does not teach the features of, "...determining a priority associated with the at least one shared web module;" and "resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any." *Croney* fails to teach or suggest the claimed features missing from the teaching of *Ramme.*

As shown above, *Ramme* does not teach or suggest the claimed features; and *Croney* fails to teach or suggest the claimed features missing from the teaching of *Ramme* therefore, combining the creating of web pages that share a common layout teaching of *Croney* cannot produce the results of the claimed invention. . Accordingly, the proposed combination of references, considered as a whole, does not teach or suggest the claimed features. The Examiner believes otherwise and cites various portions of *Croney*. Regarding claim 6, the Examiner states:

> "Regarding claim 6, most of the limitations of this claim have been noted in the rejection of claim 1. However, *Ramme* does not expressly disclose the claimed feature of "determining a priority associated with the at least one shared web module" and resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any". *Croney* et al, from the same or similar field of endeavors, discloses the server system can be configured to control which of multiple master pages are utilized to form a resulting page based upon predetermined criteria that are satisfied by the occurrence of one or more events or circumstances (see Croney et al., paragraph [0056]). Croney et al. further discloses that the modules of the server system can be configured with appropriate computer-executable instructions to recognize predetermined criteria that imposed by a master page, client system, or server system and to recognize when these predetermined criteria have been satisfied (see *Croney et al.*, paragraph [0058]). Thus, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references to achieve the claimed features of

"determining a priority associated with the at least one shared web module" and "resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any (as cited in claim 6). Such combination would have permitted teaching's *Croney* to allow Ramme's to avoid duplicating of code and content between multiple web pages and unnecessarily filling up the storage with duplicative content (see *Croney*, paragraph [0007]).

Regarding the feature, "determining a priority associated with the at least one shared web module; and resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any" the Examiner believes the following portion of *Croney* teaches this claimed feature:

> "[0056] In another embodiment, the server system can be configured to control which of multiple master pages are utilized to form a resulting page based upon predetermined criteria that are satisfied by the occurrence of one or more events or circumstances. This embodiment can be useful, for example, to enable a server system to provide different content/services to different users based upon different subscriptions, privileges, authorization levels, and client system requirements, while at the same time enabling multiple web pages to share a common layout specified by the one or more master pages.
>
> [0058] To enable these alternative embodiments, the modules of the server system can be configured with appropriate computer-executable instructions to recognize predetermined criteria that are imposed by a master page, client system, or server system and to recognize when these predetermined criteria have been satisfied.
>
> [0007] Yet another problem that can be associated with the duplicating of code and content between multiple web pages is that the storage limits of the host system can be filled-up unnecessarily with duplicative content. In other words, a large segment of content or code can undesirably strain the storage capacity of the host when the content segment is redundantly and simultaneously stored for each related web page."

Croney paragraphs [0056], [0058] page 5 and [0007] page 1.

However, *Croney* teaches away from the claimed invention by disclosing, "...to provide different content/services to different users..." This teaching stands in stark contrast with the claimed feature of, "...shared web module ..." Sharing modules by applications results in the using the same module by more than one application, and not different things for different users as taught by *Croney*. *Croney* suggests that even when using a common layout, different users may receive different content and services depending upon their circumstances. Further, the statement that, "...multiple web pages to share a common layout specified by the one or more master pages..." in *Croney* is within an application, and not shared across applications as claimed. Rather than sharing modules as recited in the claim, the master page layout of *Croney* is a constant with only the contents varying, "... upon predetermined criteria that

are satisfied by the occurrence of one or more events or circumstances..." Therefore, *Croney* teaches filling of a template that is the master page layout, and does not teach sharing of web modules as recited in the claim.

Additionally, no other portion of *Croney* teaches this claimed feature. In fact, *Croney* is unrelated to the claimed features of, "determining a priority associated with the at least one shared web module; and resolving any conflicts between shared web modules in the at least one shared web module and conflicts between the at least one shared web module and web modules of the web application, if any." Thus, *Croney* is devoid of disclosure regarding the feature of claim 6. Further, *Croney* also fails to provide the claimed elements missing from the teaching of *Ramme*. Accordingly, under the standards of, *In re KSR Int'l.* the Examiner has failed to state a *prima facie* obviousness rejection against claim 6 using *Ramme* in combination with *Croney*.

Regarding claim 8, the Examiner states:

> "Regarding claim 8, most of the limitations of this claim have been noted in the rejection of claim 1. However, *Ramme* does not expressly disclose the claimed feature of "logically merging the at least one shared web module with the web modules of the web application includes using a service provider interface (SPI) that provides merge logic for merging different module types". *Croney* et al, from the same or similar field of endeavors, discloses a user may enter commands and information into the computer through keyboard, pointing device or other input devices which are often connected to the processing unit through a serial port interface coupled to system bus. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB) (see *Croney et al.*, paragraph [0066]). Thus, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references to achieve the claimed feature of "logically merging the at least one shared web module with the web modules of the web application includes using a service provider interface (SPI) that provides merge logic for merging different module types" (as in claim 8). Such combination would have permitted teaching's *Croney* to allow *Ramme*'s to avoid duplicating of code and content between multiple web pages and unnecessarily filling up the storage with duplicative content (see *Croney*, paragraph [0007])."

Office Action August 16, 2007 pp 15-16.

Croney fails to teach the claimed feature because *Croney* fails to teach or suggest the features of claim 8. In fact, the only similarity between the teaching of *Croney* and the claimed subject matter is the term "interface." However, *Croney* fails to teach or suggest any of the other features of claim 8.

With regard to claim 8, the claim as originally provided states:

> "The method of claim 1, wherein logically merging the at least one shared web module with the web modules of the web application includes using a service

provider interface (SPI) that provides merge logic for merging different module types."

The proposed combination does not teach the features of, "...logically merging the at least one shared web module with the web modules of the web application includes using a service provider interface (SPI) that provides merge logic for merging different module types." *Croney* further fails to teach the claimed features missing from the teaching of *Ramme*. As shown above, *Ramme* does not teach or suggest the claimed features; therefore, combining the creating of web pages that share a common layout teaching of *Croney* cannot produce the results of the claimed invention.

The cited reference teaches use of hardware interfaces to connect input devices to a computer. These teachings stand in stark contrast to the programming interface, as claimed, in the form of, "a service provider interface (SPI) that provides merge logic for merging different module types." Thus, the proposed combination of references, considered as a whole, does not teach or suggest the features of the claims.

Regarding the features of, "logically merging the at least one shared web module with the web modules of the web application includes using a service provider interface (SPI) that provides merge logic for merging different module types," the Examiner believes the following portions of *Croney* teach the claimed features:

> "Program code means comprising one or more program modules may be stored on the hard disk 439, magnetic disk 429, optical disk 431, ROM 424 or RAM 425, including an operating system 435, one or more application programs 436, other program modules 437, and program data 438. A user may enter commands and information into the computer 420 through keyboard 440, pointing device 442, or other input devices (not shown), such as a microphone, joy stick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 421 through a serial port interface 446 coupled to system bus 423. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor 447 or another display device is also connected to system bus 423 via an interface, such as video adapter 448. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers."

Croney teaches use of an interface for the connection of input devices, but fails to teach, "...logically merging the at least one shared web module with the web modules of the web application includes using a service provider interface (SPI) that provides merge logic for merging different module types ..." as recited in claim 8. Thus, *Croney* is unrelated to, and devoid of disclosure regarding, the feature of claim 8. Further, *Croney* also fails to provide the claimed elements missing from the teaching of *Ramme*. Accordingly, under the standards of,

KSR Int'l. the Examiner has failed to state a *prima facie* obviousness rejection against claim 8 using *Ramme* in combination with *Croney*.

Claims 14 and 16, having all limitations of claim 6 and 8 respectively, also overcome the obviousness rejections for the reasons given above. Therefore, the rejection of claims 6, 8, 14 and 16 under 35 U.S.C. § 103 has been overcome.

C) Additionally, the examiner failed to state a *prima facie* obviousness rejection against the claims because the examiner failed to state a proper reason to achieve the legal conclusion of obviousness under the standards of KSR Int'l. Instead of offering such reasons, the examiner only offers purported advantages to combining the references. For example, the examiner states that:

> Such combination would have permitted teaching's Spotswood to allow Ramme's to better control over the reloading and namespace separation of individual modules, including EJB's (see Spotswood, paragraph [0019]).

Office action of August 16, 2007, p. 12.

The examiner makes, in structure, similar statements regarding all of the obviousness rejections. Specifically, all of these statements purport to provide an advantage to combining the references. However, though the examiner does not connect the purported advantages to reasons to achieve the *legal conclusion* of obviousness. Instead, the examiner forces the reader to *assume* that the purported advantages would somehow compel the conclusion of obviousness. Under the standards of *KSR Int'l.*, the examiner is not permitted to rely on such conclusory statements, but rather must set forth a rational underpinning to achieve the *legal conclusion* of obviousness. Accordingly, the examiner failed to state a *prima facie* obviousness rejection against the claims.

VI. Conclusion

The subject application is patentable over the cited references and should now be in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if, in the opinion of the Examiner, such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: November 16, 2007

Respectfully submitted,

/Theodore D. Fay III/

Theodore D. Fay III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants